Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СИСТЕМА ДЛЯ ЗВУКОВОГО ОФОРМЛЕННЯ ОФЛАЙН ТА ОНЛАЙН ПОДІЙ

SYSTEM FOR SOUND DESIGN OF OFFLINE AND ONLINE

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Трофімов Владислав Валерійович

(підпис)

Керівник:
к.т.н., доцент
Поліщук Микола Миколайович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 07 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Трофімову Владиславу Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система для звукового оформлення офлайн та онлайн подій*

Керівник роботи *к.т.н., доцент Поліщук Микола Миколайович*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблеми за темою роботи та постановка завдань, дослідження існуючих методів та засобів проведення дослідження, розробка й обгрунтування рекомендацій системи, оцінка результатів дослідження

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Поліщук М.М., доцент*		
Теоретичне дослідження та практична реалізація	*Поліщук М.М., доцент*		
Практична реалізація системи додаток-бот	*Поліщук М.М., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,асистент*		

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Проведення огляду літератури із досліджуваної проблеми*	до 15.02.2024 р.	Виконано
2.	*Проведення аналізу систем керування дискорд ботами та голосовими каналами*	до 15.03.2024 р.	Виконано
3.	*Розробка системи керування Discord ботами на базі Android додатку*	до 04.05.2024 р.	Виконано
4.	*Висновки та пропозиції*	до 07.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 10.05.2024 р.	Виконано
6.	*Формування додатків*	до 15.05.2024 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 20.05.2024 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2024 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2024 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 11.06.2024 р.	Виконано

Здобувач вищої освіти _____ Трофімов В.В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Поліщук М.М.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Трофімов В.В.Система для звукового оформлення онлайн зустрічей. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024. 75 с.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено огляду предметної олбасті, тут розглядаються різновиди діскорд ботів, методи інтеракції з ними. Також в цьому розділі здійснено огляд платформа для відеоконференцій Discord.

В другому розділі здійснено вибір та обгрунтування засобів розробки системи керування Discord ботом. Обрано засоби: NodeJS, Kotlin та SQL.

Третій розділ присвячено опису розробки Discord бота, мобільному Android додатку, що використовується в якості інтерфейсу для керування, та бази даних для збереження інформації мобільного додатку.

Ключові слова: бот, Discord, Android, мобільний додаток, база даних, вебхук, Raspberry PI, JavaScript, Kotlin.

ANNOTATION

Trofimov V.V. System for sound design of online meetings. Manuscript.

Qualification work for bachelor's degree in Computer Engineering, speciality 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024. 75 c.

The qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter is devoted to an overview of the subject area, it considers the types of discount bots, methods of interaction with them. This section also provides an overview of the Discord video conferencing system.

In the second section, the tools for developing a Discord bot control system are selected and discussed. The selected tools are: NodeJS, Kotlin, and SQL.

The third section describes the development of the Discord bot, a mobile Android application used as an interface for management, and a database for storing information of the mobile application.

Keywords: bot, Discord, Android, mobile application, databases, webhook, Raspberry PI, JavaScript, Kotlin.

ЗМІСТ

ВСТУП

Актуальність теми зумовлена тим, що у сучасному динамічному світі, де люди все частіше працюють та спілкуються віддалено, відеоконференції стали незамінним інструментом для ефективної онлайн-взаємодії. Одною з найпопулярніших платформ для відеоконференцій став Discord через свою зручність користування, можливість створення активних серверів для комунікації та гнучкість налаштування. Сервери Discord в сучасному інтернеті є неймовірно важливою складовою онлайн спільнот та робочої комунікації в IT компаніях. І одною з важливих причин цього є підтримка кодерів розробниками платформи, що створюють сторонні боти інтерфейси для модерації, фільтрації чату, керуванням голосовими каналами та багатьма іншими функціями. На даний момент існує більше шести тисяч написаних та перевірених ботів, які можна установити на свій сервер безпосередньо на платформі. Проте, серед подібних проектів не було знайдено ні одного бота, що має інтерфейс у вигляді мобільного додатку.

Метою роботи є розробка системи звукового оформлення онлайн зустрічей у вигляді мобільного додатку на базі Android та базою даних на Raspberry PI.

Об’єкт дослідження – інтерфейс у вигляді мобільного додатку для керування Discord ботом.

Предмет дослідження – медоти комунікації між сторонніми додатками та Discord сервером

Завдання, які необхідно виконати :

– Розробити Discord бота з стороннім інтерфейсом, мобільний додаток для взаємодії з ботом та базу даних для зберігання інформації.

– Дослідити методи взаємодії з дискорд сервером та створення діскорд ботів

– Визначити інструментарій для реалізації проекту.

– Створення бази даних на Rasberry PI

РОЗДІЛ 1
ТЕОРЕТИЧНА ЧАСТИНА
1.1 Огляд та подальший аналіз платформ для відеоконференцій

Платформи для відеоконференцій це програмне забезпечення, яке дозволяє спілкуватися візуально та за допомогою аудіо через інтернет. Також, були і апаратні рішення, які дозволяли реалізовувати віддалений відеозв'язок. Технології, що дозволяли комунікувати за допомогою відеоконференцій розвивались з п'ятидесятих років минулого століття. Першою реалізацією відеозв'язку була система «Picturephone» (рис. 1.1). Очевидно, що використовувала подібна система не інтернет, а аналоговий сигнал.

Рисунок 1.1 – Використання «Picturephone» 1964 рік [1]

В дев'яностих інтернет почав ставати все більш доступнішим. Ця технологія дозволила ентузіастам з Корнельського університету створити CU-SeeMe (рис. 1.2), один з перших інструментів для відеозв'язку через інтернет. Працювала вона на основі IP, та мала функціонал з'єднання «точка точка» та багатоточковий режим, при використанні обладнання MCU (Multipoint Control Unit). Проте, через низьку пропускну здатність мережі в ті роки, існували обмеження якості відео та аудіо.

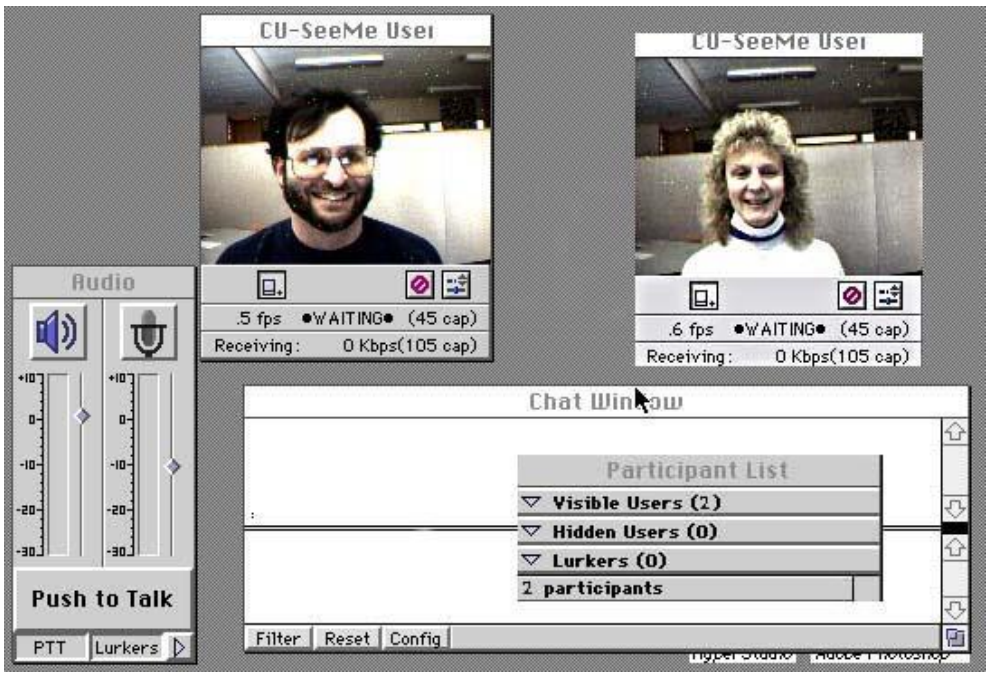

Рисунок 1.2 – Відеоконференція в CU-SeeMe [2]

В нульових свою нішу платформи для відеозв'язку зайняв Skype. Він здійснив революцію, пропонуючи безкоштовні дзвінки через інтернет та зручний, інтуїтивний інтерфейс. В свою чергу, серед корпоративних платформ лідуючим був WebEx від Cisco (рис. 1.3). Його безпекові .характеристики, та функціонал, на той час були чудовим варіантом для дистанційної роботи на той час. Ця платформа дозволяє веб-конференції до тисячі людей, та онлайн трансляції до ста тисяч користувачів.

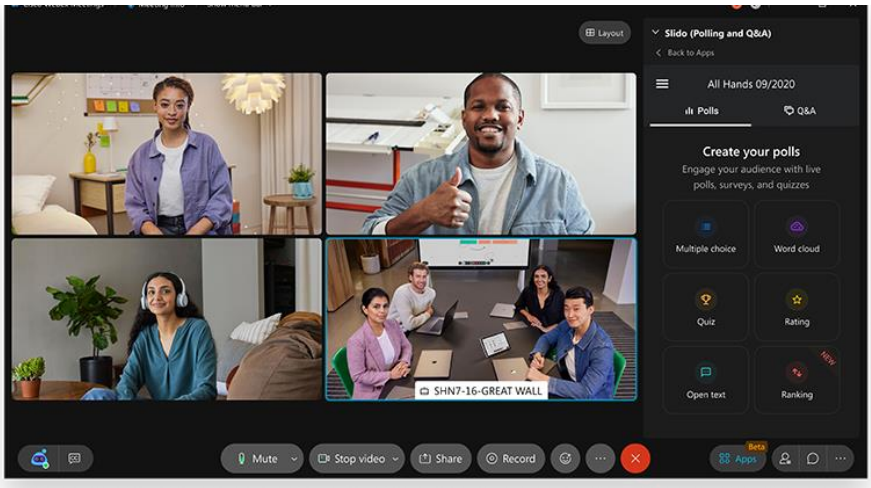

Рисунок 1.3 – Інтерфейс WebEx [3]

В період пандемії, велика кількість підприємств, освітніх та державних установ перейшла на віддалену роботу і суспільство потребувало дешевого рішення для відеоконференцій для великої кількості людей. Цю нішу зайняв Zoom, Google Meet, Microsoft Team та інші [4]. Zoom набув популярність завдяки простоті інтерфейсу та можливості запису конференцій. Ці особливості стали особливо цінними для освітніх установ, так як студенти могли переглянути урок/пару та надолужити втрачені знання. Наступні два вищезгадані представники шикоро використовуються серез підприємств через інтеграцію з іншими застосунками Google та 365 Office відповідно. Проте, з часом більшість платформ перейняли особливості один одного, тому функціонал більшості додатків зрівнявся.

Також, перед тим як перейти до Discord, варто оглянути його ідейного предка TeamSpeak (рис. 1.4). Це платформа 1999 року для аудіоконференцій, яка пропонує дуже хорошу якість зв’язку, можливість створювати канали, а також текстовий чат. Хоч в ньому відсутній відеозв’язок, проте, важливо те, що ці базові функції були в перших версіях Discord, що існувала тільки у вигляді веб-сервіса. І причиною створення останнього було те, що TeamSpeak мав важчий для розуміння інтерфейс. Попри те, що він був створений давно, досі використовується, переважно в геймерських спільнотах, де важлива мінімальна затримка звуку. Також, цей сервіс має можливість досить тонкого налаштування створених каналів комунікації. Проте, порівнюючи з Discord, в TeamSpeak написання плагінів та ботів відбувається в рази важче через те, що програмний інтерфейс відсутній. З чого випливає вузькнаправленість даного сервісу, а саме в якості голосового чату.

Підбиваючи підсумки, розвиток аудіо та відеоконференцій почався дуже давно і за цей час було реалізовано багато рішень, від апаратних до програмних. Протягом історії розвитку платформи змінювались та підлаштовувались під різні потреби соціуму. Від швидких та ефективних голосових чатів для геймерів, до відеоконференцій на сотні людей для корпоративних зібрань.

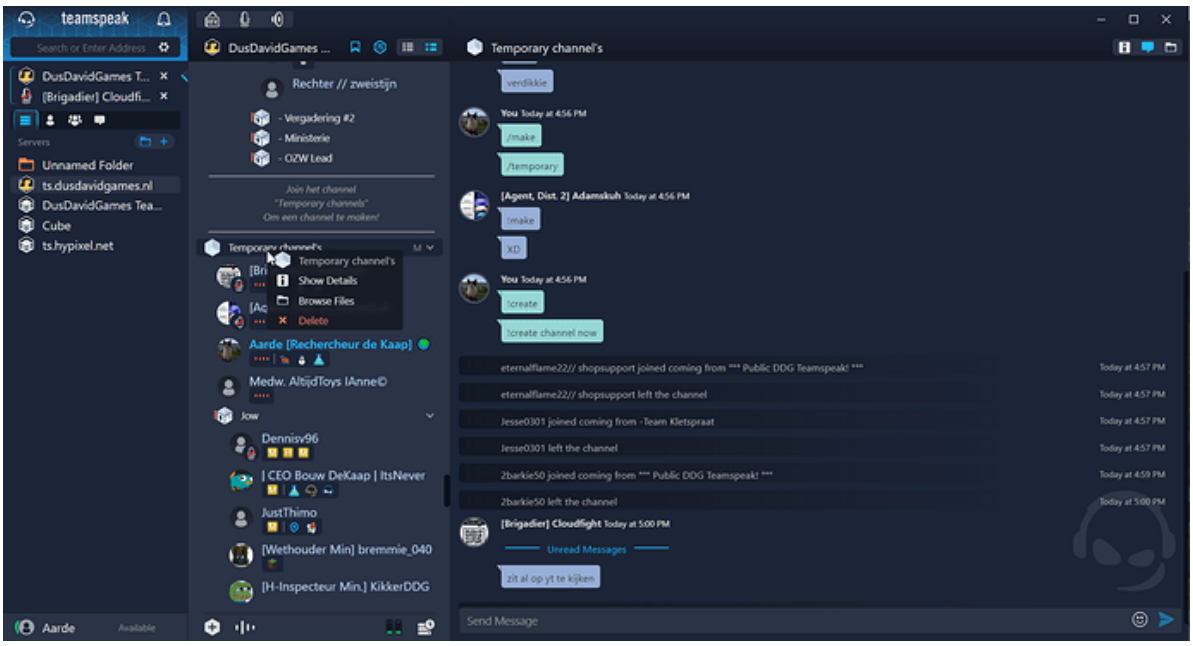

Рисунок 1.4 – новий інтерфейс TeamSpeak [5]

1.2 Огляд та характеристики Discord

Discord – це безкоштовна платформа для спілкування, доступна на різних платформах, включаючи Windows, macOS, Linux, iOS та Android. Вона здобула популярність серед різноманітних онлайн-спільнот які потребують приватної та безпечної платформи для спілкування (рис. 1.5). Discord пропонує широкий спектр функцій (табл. 1.1).

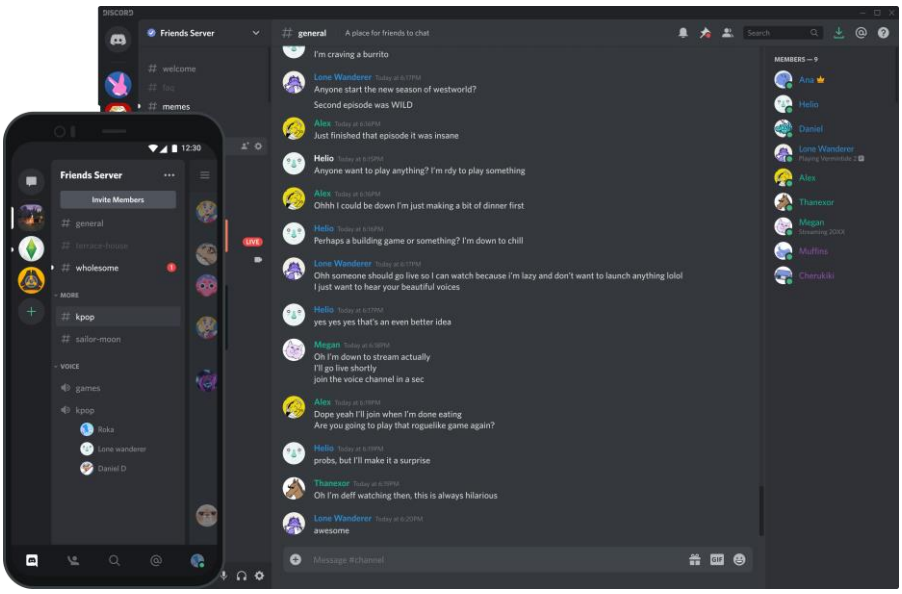

Рисунок 1.5 – Інтерфейс Discord [6]

Таблиця 1.1 – Функціонал Discord

№ п/п	Функція	Опис функції
1	Текстові повідомлення	Користувачі можуть надсилати текстові повідомлення один одному на приватних каналах або в публічних чатах.
2	Голосові та відеодзвінки	Користувачі можуть здійснювати голосові та відеодзвінки один з одним або в групах.
3	Обмін файлами	Користувачі можуть обмінюватися файлами один з одним або в групах.
4	Створення серверів	Користувачі можуть створювати власні сервери, де можуть спілкуватися з іншими людьми, які мають спільні інтереси.
5	Налаштування	Користувачі можуть налаштовувати свій профіль, сервер та інтерфейс Discord.

Discord використовує клієнт-серверну архітектуру. Це означає, що клієнтське програмне забезпечення встановлюється на комп'ютері користувача, а серверне програмне забезпечення розміщується в центрах обробки даних Discord. Клієнтське програмне забезпечення дозволяє користувачам спілкуватися один з одним, а серверне програмне забезпечення обробляє всі дані та забезпечує надійність платформи. Також воно зберігаж налаштувань та уподобань користувачів. Також, наявність мобільної версії платформи, дозволяє без перешкод користуватись додатком будь де, та використовувати його як месенджер. Discord ґрунтується на декількох теоретичних концепціях (рис. 1.6). Ці концепції, а саме відкритий код та модульність, є особливостями цієї платформи, які стануть особливо важливими для цього проекту.

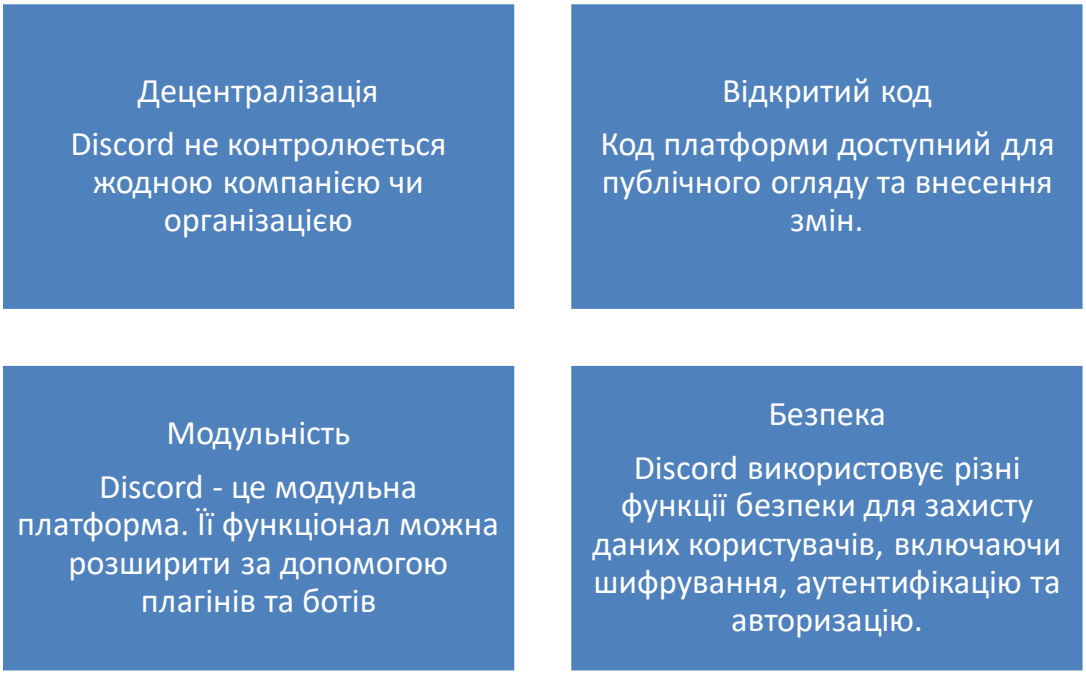

Рисунок 1.6 – Ключові концепції Discord

Основною відмінністю від інших платформ для відеоконференцій та загалом спілкування є сервери. Сервери, в контексті Discord, це віртуальні спільноти, які створюються користувачами на платформі Discord, та мають певний перелік особливостей (табл. 1.2). Вони слугують місцем для спілкування, обміну інформацією, розваг та організації людей навколо спільних інтересів. Сервери Discord можуть бути приватними або публічними, і вони можуть мати різні теми, такі як ігри, музика, спорт, наука, технології та багато іншого.

Таблиця 1.2 – Основні особливості Discord серверів

№ п/п	Характеристика	Опис характеристики
1	2	3
1	Канали	Сервери поділені на канали, які є текстовими або голосовими чатами, присвяченими певним темам
2	Ролі	Користувачам на сервері можна призначити ролі, які надають їм різні дозволи та можливості
3	Повідомлення	Користувачі можуть надсилати текстові повідомлення, зображення, відео, файли та GIF-файли в каналах

Продовження таблиці 1.2

1	2	3
4	Голосовий чат	Користувачі можуть спілкуватися за допомогою голосового чату в голосових каналах
5	Емодзі	Сервери можуть мати власні набори емодзі, які користувачі можуть використовувати у своїх повідомленнях
6	Інтеграція	Сервери Discord можна інтегрувати з іншими сервісами, такими як YouTube, Twitch та Spotify
7	Боти	На серверах Discord можна запускати ботів, які є програмами, що автоматизують певні завдання або надають додаткові функції.

Джерело: [7]

Саме цей функціонал зробив сервіс популярним в якості платформи для онлайн спільнот, адже він дозволяє без перешкод розвиватись спільноті, та нарощувати фанатську базу того чи іншого проекту. На даний момент, дуже багато незалежних розробників програмного забезпечення (ігор, додатків сервісів і тд), створюють та популяризують свої канали, щоб ділитись прогресом розробки, знаходити однодумців для допомоги і розуміти свою цільову аудиторію. Дуже часто такі незалежні проекти отримують свою користувацьку базу ще на стадії розробки продукту.

1.2 Discord API

Discord API – це набір інтерфейсів програмування застосунків (API), які дозволяють розробникам створювати програми, що взаємодіють з Discord [8]. API надає доступ до таких функцій, як:

- Створення та керування каналами та серверами.
- Надсилання та отримання текстових повідомлень.

– Здійснення голосових та відеодзвінків.

– Обмін файлами.

– Керування користувачами та ролями.

– Інтеграція з іншими програмами.

Discord API доступний для будь-кого, хто хоче його використовувати. І розробники Discord, щоб будь хто міг легко його використати у своїх потребах, адже вони створили «Портал розробника». Це онлайн-ресурс, який надає розробникам все необхідне для створення ботів та програм, що взаємодіють з сервісом Discord. Він пропонує документацію, приклади коду, інструменти та інші ресурси, які допоможуть вам розпочати роботу. Для початку роботи розробникам потрібно створити обліковий запис розробника Discord та отримати ключ API. Потім вони можуть використовувати API для створення власних програм, які взаємодіють з Discord (рис. 1.7).

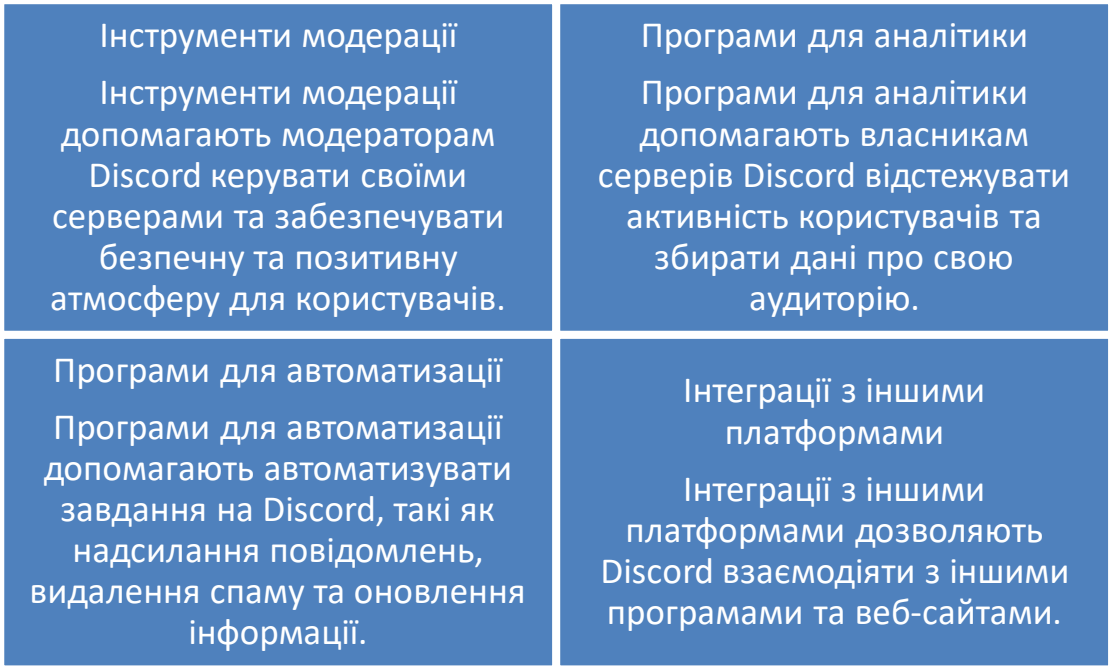

Рисунок 1.7 – Приклади та опис програм для взаємодії з Discord

Розглядаючи конкретно розробку ботів за допомогою програмного інтерфейсу Discord, можливо виділити декілька, класифікуючи їх за видом інтеракції з користувачем (рис. 1.8).

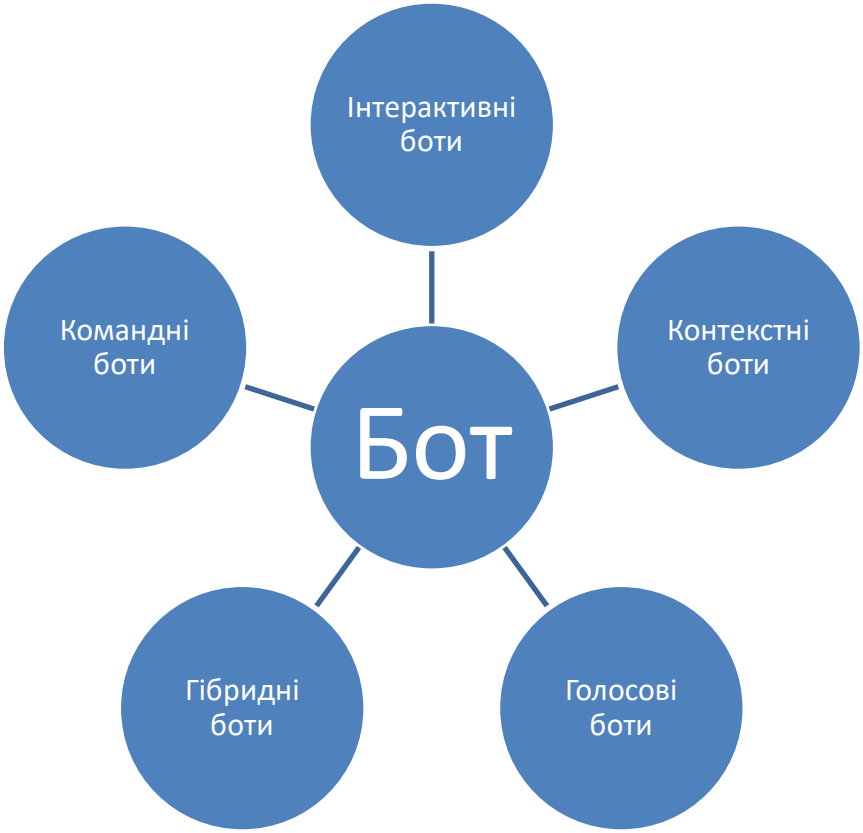

Рисунок 1.8 – Класифікація ботів за видом інтеракції

Командні боти активуються за допомогою команд, введених користувачем у чат. Використовуються для виконання різних завдань, наприклад, відтворення музики, надсилання повідомлень, або отримання інформації.

Інтерактивні боти реагують на дії користувачів, такі як натискання кнопок, вибір з меню або введення тексту. Використовуються для створення ігор, вікторин, або інших інтерактивних досвідів.

Контекстні боти аналізують текстові повідомлення користувачів та реагують на них, ґрунтуючись на контексті. Використовуються для надання інформації, перекладу мов, або генерування творчого контенту.

Гібридні боти поєднують в собі елементи командних, інтерактивних та контекстних ботів. Пропонують користувачам широкий спектр можливостей та гнучкість у спілкуванні з ботом.

Голосові боти взаємодіють з користувачами за допомогою голосових команд та відповідей. Використовуються для відтворення музики, управління розумними будинками, або надання голосових інструкцій.

Також боти можуть відрізнятись за функціоналом. Вони можуть використовуватись для автоматизації завдань модератора, надання інформації, розваг тощо (рис. 1.9) [9].

Рисунок 1.9 – Класифікація ботів за функціоналом

Одним з найпопулярніших серверів з розширеними можливостями став Midjourney. Це сервер, на якому розробники реалізували бота з інтегрованою генеративною нейромережею, яка дозволяда користувачам створювати картинки по запиту, а також збільшувати масштаб цих картинок.

РОЗДІЛ 2
ПРОГРАМНЕ ТА АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ЗВУКОВОГО ОФОРМЛЕННЯ ОНЛАЙН ЗУСРІЧЕЙ

2.1 Проектування Discord бота

Розробка Discrord ботів можлива на багатьох мовах програмування, проте, при виборі мови для реалізації потрібно спиратись на два основних фактора: бібліотеки для розробки, та спільноту розробників. Наявність бібліотек полегшить розробку Discord додатків. Чим більше бібліотек, тим більша імовірність знайти активні бібліотеки, що зменшить імовірність конфліктів при реалізації. Велика спільнота розробників дозволить знайти модифіковані варіанти бібліотек під конкретні потреби, а також полегшить відладку проекту, бо при зіткненнями з помилками в компіляторі буде імовірніше знайти вирішення проблеми. Перебираючи ці фактори найкращим варіантом виявиться JavaScript.

JavaScript це мова програмування, що зазвичай використовується для створення динамічного, інтерактивного контенту для веб-сторінок [10]. Її можна використовувати для додавання анімації, ігор, реагування на події користувачів та багато іншого. Судячи з вищеописаного, бачимо, що JavaScript вузькоспеціалізована мова програмування.

Щоб мати можливість писати серверні скрипти на JavaScript, потрібно використати спеціалізоване середовище виконання Node.js. Він дозволяє розробляти веб-сервери, мережеві програми та інше. Як бачимо, можливість створення веб-сервера дозволяє обробляти інформацію в реальному часі, що і робитиме бот [11].

Для роботи з Discord API, потрібно обрати модуль для Node.js. Discord.js стане одним з найкращих варіантів по причині його популярності, простоті інтерфейсу, відкритому коду, організації коду, за допомогою об’єктно орієнтованому програмуванню та широкого спектру функцій (табл. 2.1).

Таблиця 2.1 – Функції Discord.js

№ п/п	Функції	Опис функцій
1	Керування сповіщеннями	Створення та надсилання текстових повідомлень, зображень, відео та інших файлів у канали та приватні чати
2	Керування каналами	Створення, видалення, редагування та керування каналами, ролями та дозволами на серверах Discord.
3	Обробка подій	Реагування на дії користувачів, такі як надсилання повідомлень, клацання на кнопки та введення команд.
4	Створення інтерактивних елементів	Розробка вбудованих повідомлень, кнопок, меню та інших інтерактивних елементів для Discord.
5	Отримання та оновлення даних	Доступ до та оновлення інформації про користувачів, канали, сервери та інші дані Discord.

Джерело: [12]

Також для реалізації бота потрібне середовище розробки. Серед всіх опцій, було вибрано PHP Storm. Це високопродуктивне інтегроване середовище розробки від JetBrains. І хоч воно було спеціально розроблено для PHP-розробників, її гнучкі функції, інструменти та підтримка синтаксису інших мов програмування, JavaScript в тому числі, роблять її фаворитом, пропонуючи безліч переваг для створення, тестування та підтримки веб-додатків [13].

Для того, щоб реалізувати бота, що буде програвати аудіо, потрібно щоб він мав джерело контенту. Для цього існує ytpl модуль. Це модуль, що дозволяє підвантажувати контент з YouTube [14]. З його допомогою, можливо, за наявності посилання, або ж назви відео, отримати повну інформацію про нього.

Для реалізації стороннього інтерфейса можливо використати два варіанти:

– Створення сервера посередника на який будуть приходити POST запити від додатку, а він в свою чергу відправлятиме їх боту. Подібна реалізація можлива, проте, судячи з моїх тестових спроб, для коректної роботи сервер має знаходитись на сторонньому хостингу зі стабільним з'єднанням. Це чудовий варіант для готового продукту, через свою гнучкість, але хорощий хостинг потребує оплати.

– Вебхук це HTTP-зворотний виклик, який дозволяє одному веб-додатку надсилати дані іншому веб-додатку [15]. Це простий, але потужний спосіб для веб-додатків взаємодіяти один з одним. На сервері Discord є можиливість створити вебхук, який буде зв'язаний з цим сервером. Його можна вказати як адрес, на який буде приходити запис з додатку. Він автоматично поститиме його в текстовий час сервера. В порівнянні з минулим методом, дуже простий у використанні інструмент. Проте його мінусом є неможливість додаткового налаштування. Як результат, він засмічуватиме чат, в який будуть приходити повідомлення. Проте, це вирішується створенням спеціалізованого чату для бота.

Вивчаючи Discord ботів, а саме аудіо/музичних ботів, потрібно зазначити такі основні функції-команди, що мусять бути реалізовані:

– /play «посилання» – команда, що дозволяє боту приєднатись до голосового каналу, та програвати аудіо.

– /stop – команда, що ініціює завершення команди /play та вихід з голосового чату.

2.2 Проектування мобільного додатку на Android

Платформою для розробки додатку є Android, через поширеність операційної системи, відкритого коду та безкоштовного і зручного середовища розробки Android Studio [16]. Це офіційний інструмент розробки, що використовується Google, і воно пропонує широкий спектр функцій та

можливостей, які допомагають розробникам створювати високоякісні, потужні та інноваційні програми (табл. 2.2).

Таблиця 2.2 – Функції та особливості Android Studio

№ п/п	Функції	Опис функцій
1	Багатоплатформність	Android Studio доступне для Windows, macOS та Linux, що робить його доступним для розробників на будь-якій платформі.
2	Швидкість та гнучкість	Android Studio використовує віртуальну машину на основі Gradle, яка забезпечує швидке та гнучке середовище розробки.
3	Інструменти для дизайну інтерфейсу	Android Studio пропонує потужні інструменти для дизайну інтерфейсу, які дозволяють розробникам створювати красиві та зручні інтерфейси користувача. Як приклад, система створення верстки drag-and-drop
4	Відладчик	Android Studio має вбудований відладчик, який допомагає розробникам знаходити та виправляти помилки в своїх програмах.
5	Профілювання	Android Studio пропонує інструменти профілювання, які допомагають розробникам оптимізувати продуктивність своїх програм.
6	Підтримка Git	Android Studio має вбудовану підтримку Git, що полегшує керування кодом та співпрацю з іншими розробниками.
7	Спільнота	Android Studio має велику та активну спільноту розробників, які готові допомогти та поділитися своїми знаннями

Мовою розробки було вибрано Kotlin, завдяки тому, що Kotlin код значно коротший та читабельніший, ніж Java код [17]. Це робить його більш приємним для написання та обслуговування коду. Також він має жорстку систему статичної типізації, яка допомагає запобігти помилкам під час виконання. Це робить код Kotlin більш надійним та стійким до помилок.

Kotlin код може без проблем взаємодіяти з Java кодом. Це робить його ідеальним вибором для розробки нових функцій у існуючих Java-проектах. Він підтримує функціональне програмування, що дозволяє писати більш чіткий та елегантний код. Також Kotlin має функцію «нульові безпечні посилання», яка допомагає запобігти помилкам NullPointerException. Окрім цього, Kotlin можна використовувати для розробки Android-додатків, веб-застосунків, серверних програм та багато іншого.

Основними бібліотеками для мережевої комунікації відібрано okhttp та gson. В даному проекті вони потрібні для створення та виконання http запитів Okhttp це високопродуктивний HTTP-клієнт з відкритим кодом для Android [18]. Він пропонує ряд особливостей, які роблять його кращим вибором, ніж стандартний HTTP-клієнт Android (рис. 2.1).

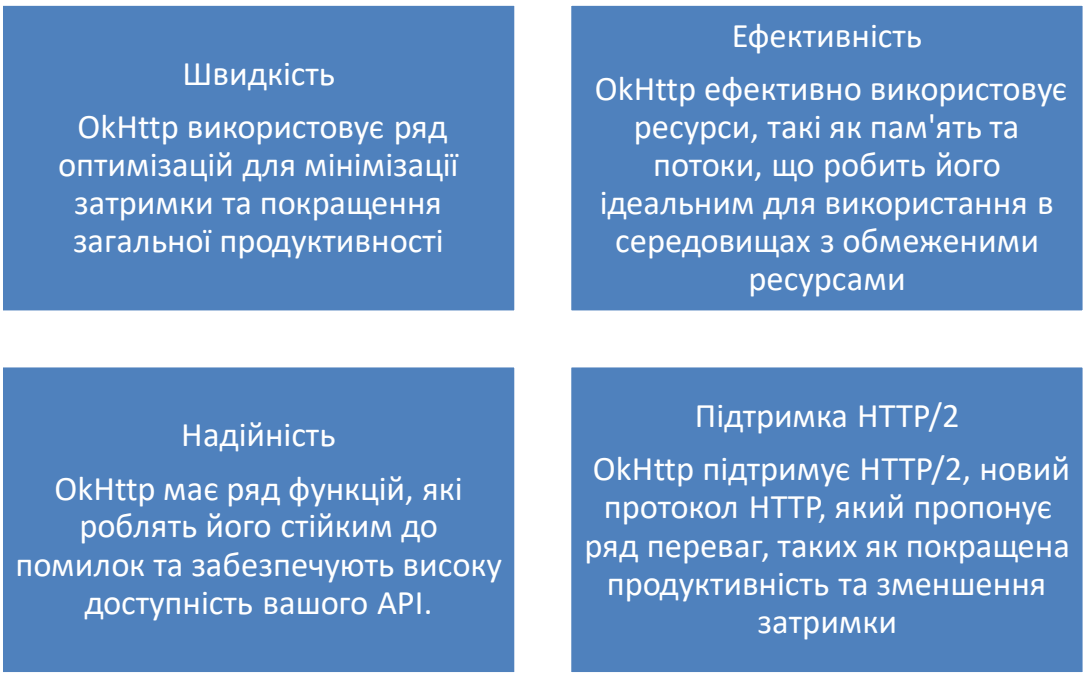

Рисунок 2.1 – Особливості бібліотеки Okhttp

GSON – це популярна бібліотека Java для перетворення JSON-даних на об’єкти Java та навпаки [19]. Вона використовується в широкому колі застосунків, включаючи розробку Android, завдяки своїй простоті, гнучкості та продуктивності. Gson може автоматично перетворювати JSON-дані в об’єкти Java, відповідно до структури JSON і навпаки, об’єкти Java в JSON, що робить їх зручними, для передачі та зберігання.

Gson підтримує широкий спектр типів даних, включаючи primitives, arrays, lists, maps, and custom objects. Також Gson можна налаштувати для форматування JSON-даних відповідно до ваших потреб.

Для взаємодією з віддаленою базою данних відібрано бібліотеки HikariCP та MariaDB. Вони використовуватимуться для взаємодії з базою даних MariaDB та оновлення додатку. HikariCP – це популярний пул з’єднань для Java, який використовується для покращення продуктивності та масштабованості при роботі з базами даних [20]. Він пропонує ряд функцій, які роблять його кращим вибором, ніж інші пули з’єднань.

HikariCP має невеликий розмір коду, що робить його ідеальним для вбудовування в різні проекти. Він використовує ряд оптимізацій для мінімізації затримки та покращення загальної продуктивності.

Також HikariCP ефективно використовує ресурси, такі як пам’ять та потоки, що робить його ідеальним для використання в середовищах з обмеженими ресурсами. HikariCP має ряд функцій, які роблять його стійким до помилок та забезпечують високу доступність вашої бази даних.

MariaDB Java Client це офіційний драйвер JDBC для MariaDB, реляційної бази даних з відкритим кодом, яка є сумісною з MySQL [21]. Він дозволяє вашому Java-додатку підключатися до бази даних MariaDB, виконувати SQL-запити та отримувати результати. За допомогою цього драйвера легко встановити з’єднання з базою даних MariaDB, він постійно обновляється, має відкритий код та дуже добре оптимізований. Велика спільнота користувачів дуже спрощує відладку додатків, що використовують подібні бази даних. Це все робить MariaDB та її драйвер, фаворитом для використання в проекті.

2.3 Проектування бази даних

MariaDB – це реляційна система керування базами даних з відкритим кодом, яка є сумісною з MySQL [22]. Вона є популярним вибором для широкого кола застосунків, завдяки своїй надійності, масштабованості та гнучкості (рис. 2.2).

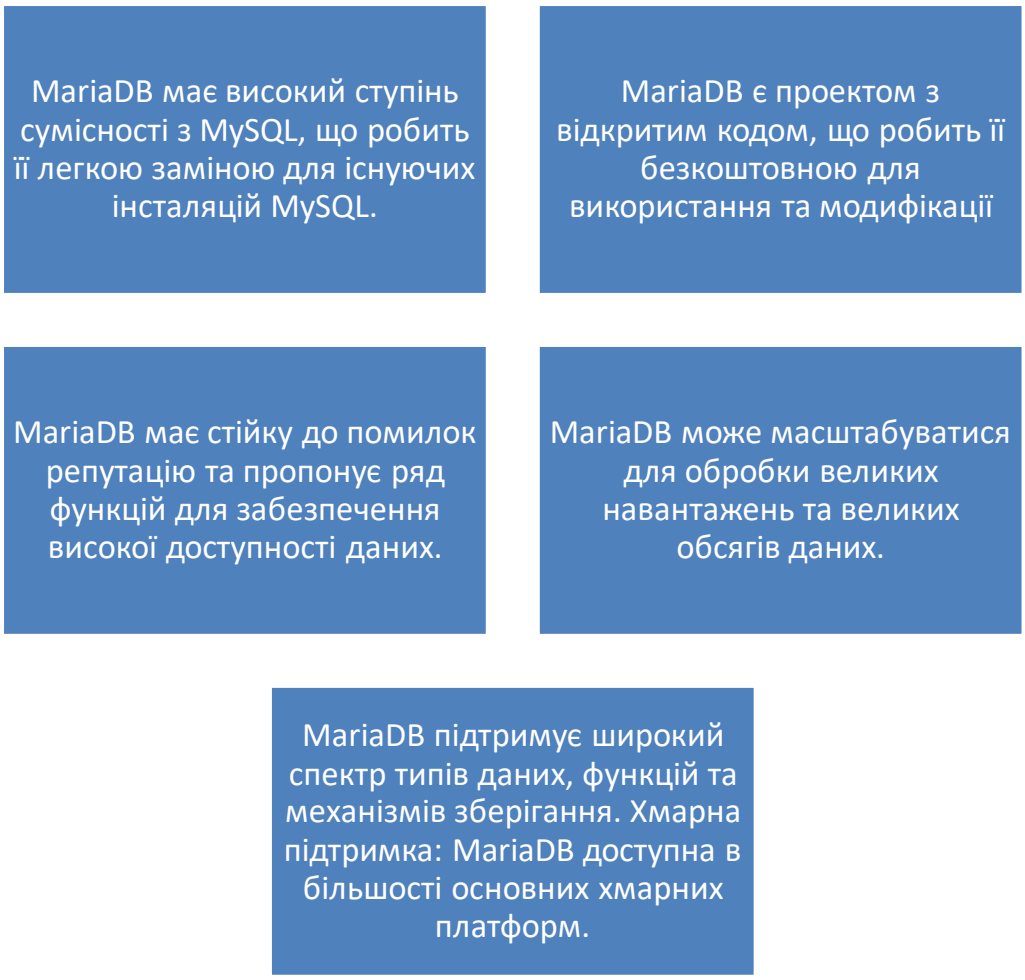

Рисунок 2.2 – Можливості MariaDB

Для керування базою даних використовувся HeidiSQL. Це безкоштовний та простий у використанні інструмент з відкритим кодом для керування базами даних, який дозволяє вам взаємодіяти з базами даних MariaDB та MySQL [23]. Він є популярним вибором серед розробників та адміністраторів баз даних завдяки своєму зручному інтерфейсу та широкому набору функцій (табл. 2.3).

Таблиця 2.3 – Функція HeidiSQL

№ п/п	Функції	Опис функцій
1	Підтримка MariaDB та MySQL	HeidiSQL підтримує як MariaDB, так і MySQL, що робить його універсальним інструментом для роботи з цими популярними системами керувамння базами даних
2	Зручний інтерфейс	HeidiSQL має інтуїтивно зрозумілий графічний інтерфейс користувача, який дозволяє легко переглядати, редагувати та керувати даними, таблицями, користувачами та дозволами
3	Запити SQL	HeidiSQL дозволяє вам виконувати запити SQL безпосередньо в інтерфейсі, що робить його зручним для роботи з даними та структурою бази даних
4	Імпорт та експорт даних	HeidiSQL дозволяє вам імпортувати та експортувати дані бази даних у різних форматах, таких як CSV, SQL та XML
5	Візуальний редактор таблиць	HeidiSQL дозволяє редагувати структуру таблиць за допомогою візуального редактора, що є більш зручним, ніж використання SQL-запитів. Безпека: HeidiSQL підтримує безпечні з’єднання з базами даних за допомогою шифрування

Домашній сервер для бази даних тестувався на персональному комп’ютері та на платі Raspberry Pi 4. Raspberry Pi 4 Model B це одноплатний комп’ютер, розроблений Raspberry Pi Foundation [24]. Це четверта ревізія Raspberry Pi і найпотужніша на сьогодні. Він має чотири ядра Broadcom BCM2711 з тактовою частотою 1,5 ГГц, 4 ГБ оперативної пам’яті LPDDR4-3200, два порти micro-HDMI, що підтримують роздільну здатність до 4K при 60 кадрах в секунду, два порти USB 3.0, два порти USB 2.0, Gigabit Ethernet, порт камери CSI та порт дисплея DSI. Цей одноплатний комп’ютер можна для різних цілей, зокрема для навчання програмуванню, як медіацентр, для створення ретро-ігр та робототехніки, як пристрій інтернету речей та веб сервер. У цьому випадку він використовуватиметься саме як сервер.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ ЗВУКОВОГО ОФОРМЛЕННЯ ОНЛАЙН ЗУСТРІЧЕЙ

3.1 Реалізація діскорд бота

Створення бота для дискорду передбачає в першу чергу його реєстрацію. Для цього портрібно зайти на портал розробників Discord та знайти кнопку New Application. Надалі потрібно назвати бота та перейти на сторінку налаштування. В параметрах Bot потрібно знайти токен. Його можна отримати тільки один раз, наступний буде генеруватись заново. Також потрібно впевнитись, про функції PRESENSE INTENT, SERVER MEMBER INTENT та MESSAGE CONTENT INTENT активні Налаштування проекту опції bot можна побачити на рисунку 3.1.

Далі потрібно перейти по опції OAuth2, для отримання client ID та створення посилання для додавання бота на сервер. Для створення посилання задієна механіка прапорців. Обов'язково вказати redirect, тобто базу генерації посилання. Проект аудіо Discord бота потребує активаці двох основних прапорців bot та voice, що визначатимуть спеціалізацію додатка. В дозволах бота, найкраще надати права Administrator, особливо на етапі розробки додатка. Генерацію посилання можна побачити на рисунку 3.2.

Для тестування бота варто створити свій сервер. До моменту перевірки додатку, добавити на сервер його можливо тільки розробнику. Для цього потрібно перейти по згенерованому посиланню. Проте, коли сервер пройде перевірку та відмітку в сто різних серверів, що активно його використовують, його розмістять в меню додатків Discord. Вигляд сторінки приєднання бота зображено на рисунку 3.3. Також, варто відмітити, що сервери, сервіс, що він пропонує можливо монетизувати. Гарним прикладом цього є вищезгаданий в підрозділі 1.3 MidJourney, який спочатку пропонував підписку, для швидшої та якіснішої генерації контенту, а в подальшому повністю перейшов до системи

платного використання платформи. Монетизація серверу, та наданого сервісу відбувається автоматизовано на самій платформі Discord.

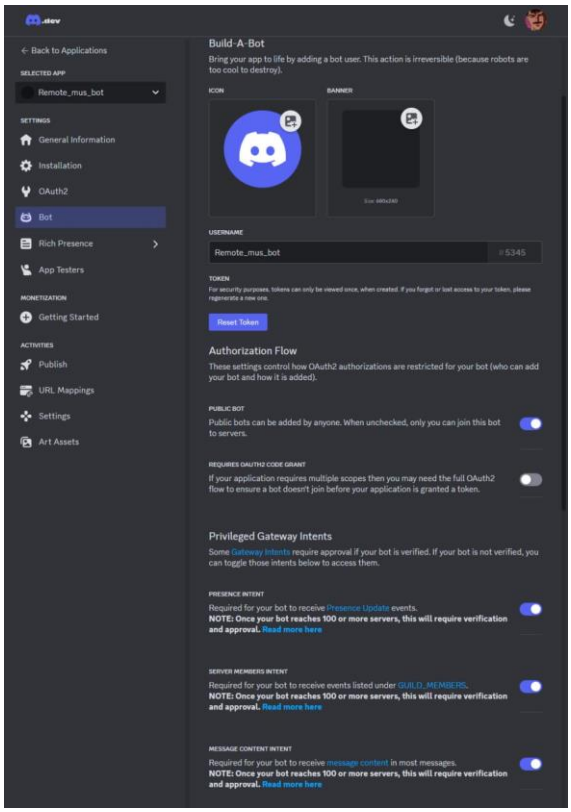

Рисунок 3.1 – Параметри опції bot

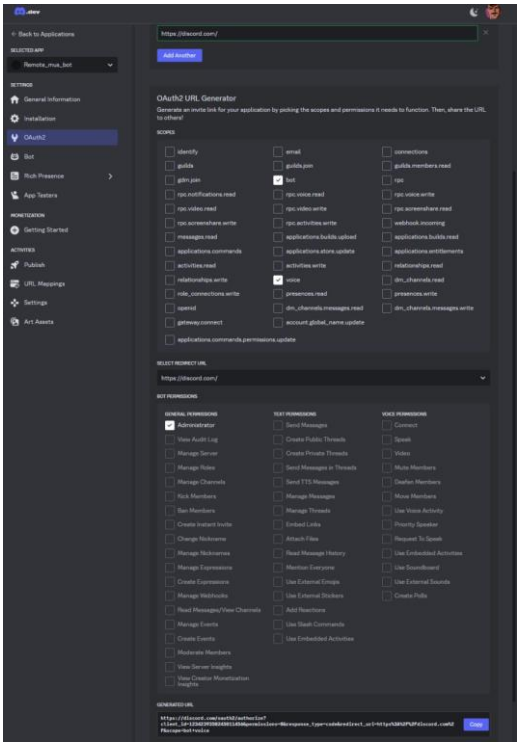

Рисунок 3.2 – Генерація посилання для додавання бота

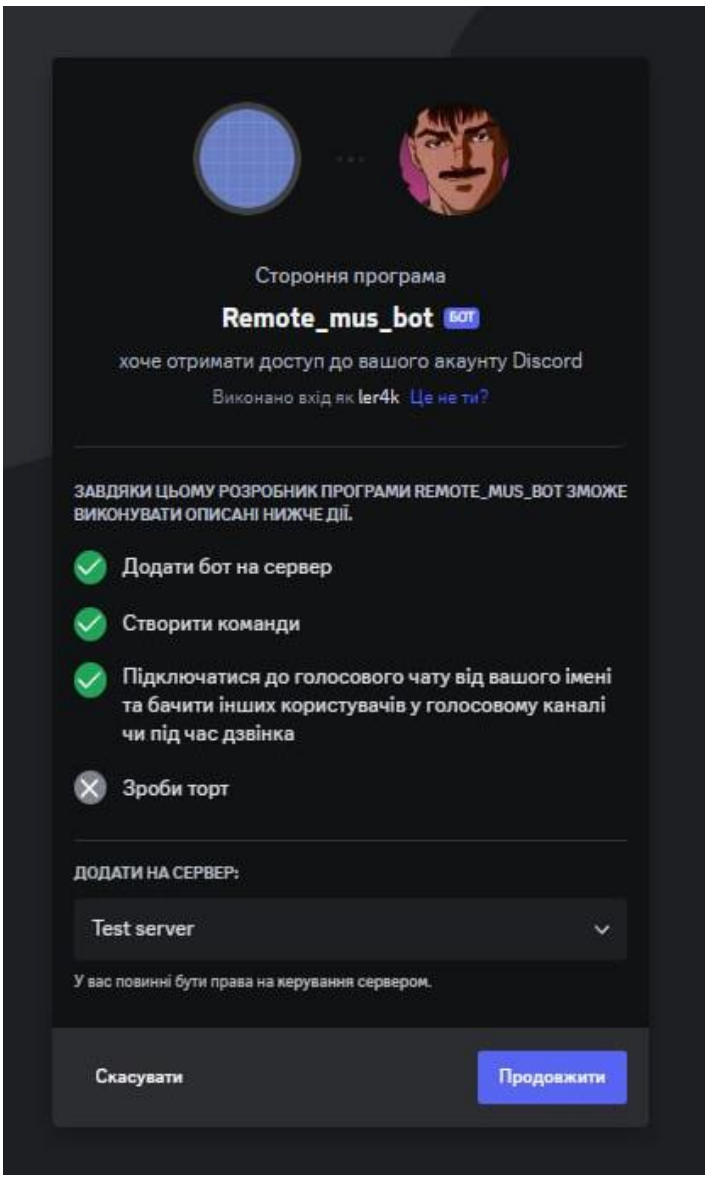

Рисунок 3.3 – Додавання бота на сервер

Після натискання на кнопку продовжити, з’являється запит на отримання дозволів адміністратора. Ми вже налаштували дозволи, тому просто авторизуємо бота, та переходимо на сервер для перевірки. Переглянути дозволи бота можиливо в параметрі інтеграцій сервера. Дозволи бота на сервері описані на рисунку 3.4. Варто зазначити, що система дозволів розповсюджується також на користувачів. Як вже вище зазначалось, Discord має функції ролей. Ці ролі потрібні для автоматизації видачі дозволів користувачам, які займають посаду адміністратора, модератора або ж будь яку іншу, що потребує сервер. Це дозволяє створювати певну ієрархію та спрощувати процес адміністрування.

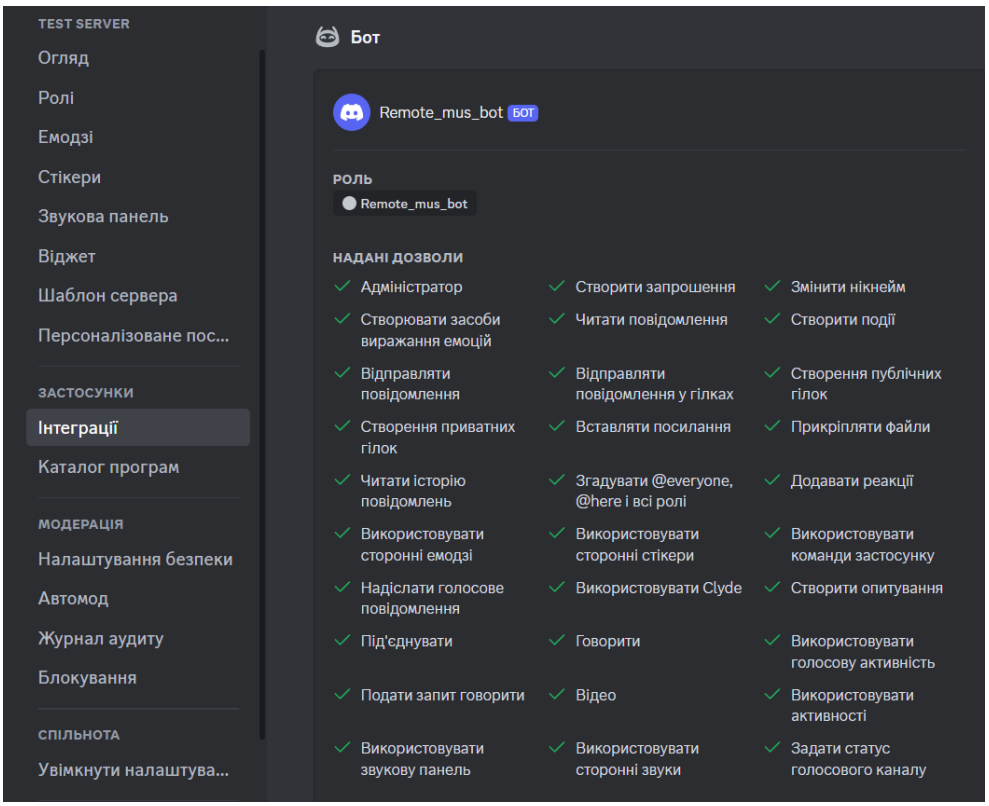

Рисунок 3.4 – Дозволи бота

Також потрібно отримати ключ бота, який можна знайти на головній сторінці додатка на порталі розробників, та зв’язати локальну машину з Discord API, через цей ключ, щоб мати змогу запускати бота через SSH та термінали серидовищ розробки. На рисунку 3.5 зображено знаходження ключа. Сам ключ закрито, для безпеки проекту.

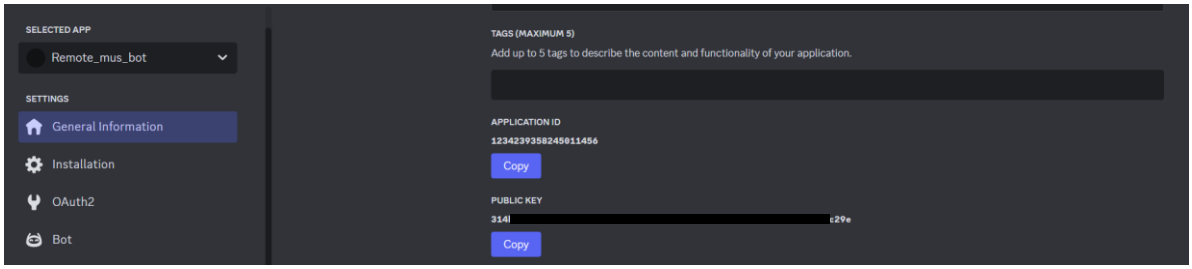

Рисунок 3.5 – Росташування ключа додатку

Наперед створимо webhook, для взаємодії додатку та бота. Для цього потрібно зайти на сервер, вибрати конкретний канал, для інтеграції вебхука, та натискаємо на кнопку «Новий вебхук» (рисунок 3.6).

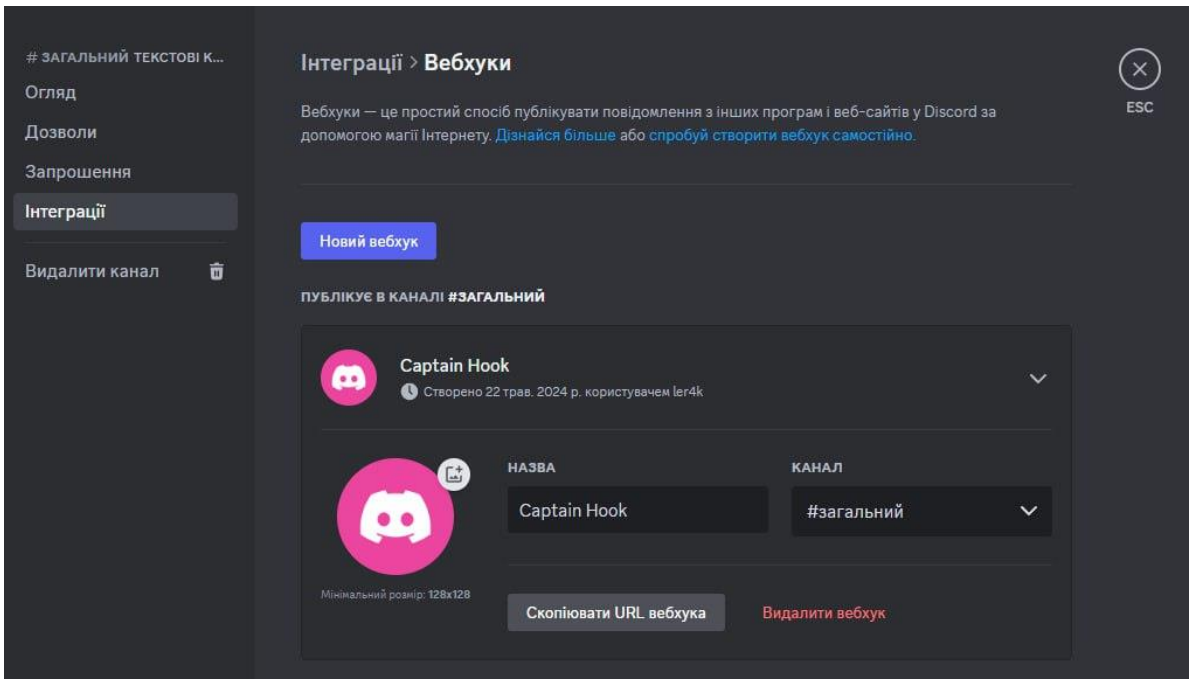

Рисунок 3.6 – інтерфейс створення та редактування вебхука

Структура проекту проста, та складається з index.js файлу, файлу зі змінними .env, та json файли маніфесту, необхідні для роботи проекту.Оглянути її можна на рисунку 3.7.

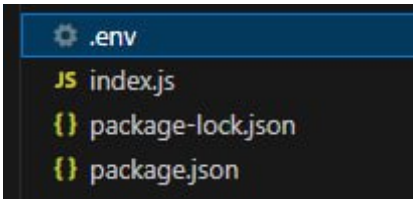

Рисунок 3.7 – Файлова структура бота

Оглянемо код файлу package.json. Цей маніфест містить основну інформацію, а саме назва проекту, його опис, основний файл проекту, версію Node.js, а також залежності, які підгружаються за допомогою консолі середовища розробки, або ж SSH (лістинг 3.1). Він створюється автоматично, при встановленні Node.js на проект, та потрібен для функціонування веб-серверу, чи будь-якої іншої функції проекту. Без цього файлу, не може сформуватись package-lock.json, який і використовується при «підніманні» проектів.

Лістинг 3.1 – Package.json (інформація про проект)

```
{
  "name": "test",
  "displayName": "rem_mus_bot",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "engines": {
    "node": "18.x"
  },
  "dependencies": {
    "@discordjs/opus": "^0.9.0",
    "@discordjs/voice": "^0.14.0",
    "discord.js": "^14.11.0",
    "dotenv": "^16.0.3",
    "node-opus": "^0.3.3",
    "opusscript": "^0.1.1",
    "tweetnacl": "^1.0.3",
    "ytdl-core": "^4.11.5",
    "ytpl": "^2.3.0"
  }
}
```

Кінець лістингу 3.1

Перейдемо до детального огляду файлу index.js. Перше, що варто описати це обробка подій та налаштування статусу присутності для бота (лістинг 3.2).

Лістинг 3.2 – Прослуховування подій та їх обробка методом bot.on()

```
bot.on("warn", console.warn);
bot.on("error", console.error);
bot.on("ready", () => console.log(bot.user.tag + " Ви отримали доступ до свого облікового запису."));
bot.on("shardDisconnect", (event, id) => console.log("Shard " + id + " disconnected (" + event.code + ") "
+ event + ", trying to reconnect..."));
bot.on("shardReconnecting", (id) => console.log(" Shard " + id + " reconnecting..."));
bot.on("ready", () => {
   bot.user.setPresence({
      status: "idle"
   });
});
```

Кінець лістингу 3.2

– bot.on("warn", console.warn) це рядок налаштовує слухач для події "warn". Коли виникає попередження, пов’язане з роботою бота, викликається функція console.warn, яка записує повідомлення про попередження в консоль.

– bot.on("error", console.error). Аналогічно до попереднього рядка, цей рядок налаштовує слухач для події "error". Якщо під час роботи бота виникають помилки, викликається функція console.error, яка записує повідомлення про помилку в консоль для налагодження.

– bot.on("ready", () => console.log(bot.user.tag + " Ви отримали доступ до свого облікового запису.")) це рядок відстежує подію "ready". Коли бот успішно підключається до Discord і готовий приймати команди та взаємодіяти з сервером, виконується ця функція. Вона записує в консоль повідомлення, яке включає ім’я користувача та тег бота, а також українське повідомлення про те, що він успішно увійшов.

– bot.on("shardDisconnect", (event, id) => console.log("Shard " + id + " disconnected (" + event.code + ") " + event + ", trying to reconnect...")) це рядок відстежує подію "shardDisconnect". Discord використовує концепцію шардування для обробки великої кількості користувачів і серверів. Ця подія спрацьовує, коли шард (субпроцес, який керує частиною функціональності бота) відключається від Discord. Код записує інформацію про від’єднаний шард, включаючи його ID, код від'єднання, а також намагається записати весь об’єкт події для подальшого аналізу. Він також записує повідомлення про те, що бот намагається повторно підключитися.

– bot.on("shardReconnecting", (id) => console.log(" Shard " + id + " reconnecting...")) Цей рядок відстежує подію "shardReconnecting". Якщо від’єднаний шард намагається повторно підключитися до Discord, спрацьовує ця подія. Код просто записує повідомлення про те, який шард повторно підключається.

Налаштування присутності: bot.on("ready", () => { ... }) Цей блок коду виконується в рамках слухача події "ready".

– bot.user.setPresence({ status: "idle" }) Цей рядок встановлює статус присутності бота на "idle". Це повідомляє користувачам, що бот онлайн, але не активно зайнятий.

Також, варто зазначити, що реалізована обробка невиконаних відхилень обіцянок (лістинг 3.3).

Лістинг 3.3 – обробка невиконаних обіцянок

```
process.on("unhandledRejection", (error) => {
   console.error("Unhandled promise rejection : ", error);
});
```

Кінець лістингу 3.3

Обробка необроблених відхилень обіцянок має вирішальне значення для стабільності Discord-бота. Це гарантує, що помилки не залишаться непоміченими і не призведуть до несподіваних проблем. Реєструючи помилку, можливо визначити першопричину відхилення та виправити основну проблему у коді.

Оглянемо функції, які використовуються при обробці запитів з вебхука. Через те, що запит публікується в текстовому каналі, а створення сповіщення вебхуком не вважається подією інтеракції, потрібно відділити посилання на контент, та команду, яка вказує на вид взаємодії з ботом. Для цього реалізована функція extractYoutubeLink (лістинг 3.4).

Лістинг 3.4 – функція extractYoutubeLink

```
function extractYouTubeLink(messageContent) {
   const cleanedURL = messageContent.replace("/play ", "");

   const regex = /https?:\/\/(www\.)?youtu\.(be|be\/|watch\?v=)?([\w-]+)/;
   const match = messageContent.match(regex);
   console.log(cleanedURL)
   if (match) {
      return cleanedURL;
   }
   return null;
}
```

Кінець лістингу 3.4

Очищення вмісту повідомлення досягається методом const cleanedURL = messageContent.replace("/play ", ""). Цей рядок видаляє префікс "/play " із вмісту повідомлення. Цепотрібно, щоб гарантувати, що функція зосереджена лише на фактичному посилання YouTube, незалежно від того, як користувач сформулював команду.

– const regex = /https?:\/\/(www\.)?youtu\.(be|be\/|watch\?v=)?([\w-]+)/: визначає регулярний вираз (regex), який намагається відповідати дійсним форматам посилань YouTube. Ось розбивка шаблону:

– https?://: Відповідає протоколу http або https (необов'язково). (www\.)?: Відповідає необов’язковій частині "www." доменного імені.

– youtu\.: Відповідає частині «youtu.» домену YouTube.

– be: Відповідає TLD (Top-Level Domain)

– watch\?v=: Відповідає частині «watch?v=» URL-адреси, за якою йде знак рівності. ([\w-]+): Відповідає одному або декільком символам слова (\w) та дефісам (-), щоб захопити ID відео.

– const match = messageContent.match(regex) зіставляє регулярний вираз (regex) із рядком вмісту повідомлення. Якщо відповідність знайдена, вона буде збережена у змінній match.

– if (match) перевіряє, чи є відповідність (match не null).

– return cleanedURL відповідає за повернення ощищеної URL-адресу

– return null функція повертає null якщо відповідність не знайдена, що свідчить про те, що у вмісті повідомлення не знайдено дійсного посилання YouTube.

Також потрібно реалізувати звичну для більшості аудіо ботів можливість показу назви контенту, та його прев’ю. Вона потрібна, з декількох причин. Перша базується на тому, що користувач мусить бачити, чи правильне посилання було застосовано до бота. Друга випливає з першої, якщо декілька користувачів користуються ботом, це означає, що вони муситимуть бачити, який контент програється на даний момент. Також це слугуватиме для цього проекту елементом відладки, адже посилання ми вписуємо не напряму в чат, а

відправляємо зі стороннього сервісу, що означає, що показ контенту на відео дозволить зрозуміти, чи правильний контент зберігається в елементі списку. Для цього потрібно отримувати інформацію про контент за посиланням. Для цього реалізовано функцію getVideoInfo (лістинг 3.5).

Лістинг 3.5 – функція getVideoInfo

```
async function getVideoInfo(youtubeLink) {
    try {
        const videoInfo = await ytdl.getBasicInfo(youtubeLink);
        console.log(`ytdl.getBasicInfo(youtubeLink)`);
        return {
            url: videoInfo.videoDetails.videoUrl,
            title: videoInfo.videoDetails.title,
        };
    } catch (error) {
        console.error("Error getting video info:", error);
        return null;
    }
}
```

Кінець лістингу 3.5

– const videoInfo = await ytdl.getBasicInfo(youtubeLink) використовує await для очікування асинхронного виклику.

– ytdl.getBasicInfo(youtubeLink). є частиною бібліотеки ytdl-core та отримує базову інформацію про відео YouTube за допомогою наданого посилання.

– videoInfo. url: videoInfo.videoDetails.videoUrl витягує повне URL-адресу відео YouTube з властивості videoDetails об’єкта

– videoInfo. title: videoInfo.videoDetails.title витягує назву відео з властивості videoDetails об’єкта videoInfo.

Саме виконання функції створення голосового потоку на канал та програвання аудіо відбувається в частині коду, що відповідає за прослуховування подій створення сповіщень. В підрозділі 2.1 описувалась проблема, що вебхуки не вміють створювати активності, при публікації сповіщень, тому приходиться прослуховувати весь текст сповіщень. Варто висвітлити те, що в переважній більшості ботів, використовується саме

створення пулу команд, які вже прослуховуються як події. Це чудова практика для створення інтеракцій в ботах та в рази зручніша. Проте, у випадку цього проекту це вирішення не підходить. Перевірка інформації з запиту, та її перетворення (лістинг 3.6).

Лістинг 3.6 – Перша частина блоку bot.on(Events.MessageCreate)

```
bot.on(Events.MessageCreate, async (message) => {
    if (message.content.startsWith("/play")) {
      const youtubeLink = extractYouTubeLink(message.content);
      if (!youtubeLink) {
        message.channel.send("Invalid YouTube link.");
        return;
      }
      const videoInfo = await getVideoInfo(youtubeLink);
      if (!videoInfo) {
        message.channel.send("Unable to get video information.");
        return;
      }
      const TguildId = "1222338279555465276"
      const guild =  bot.guilds.cache.get(TguildId);
      console.log("guild " + guild);
```

Кінець лістингу 3.6

– bot.on(Events.MessageCreate, async (message) => {...}) налаштування обробника події Events.MessageCreate. Щоразу, коли надсилається повідомлення на Discord-сервер, функція в фігурних дужках буде виконуватися асинхронно.

if (message.content.startsWith("/play")) це блок, що перевіряє, чи вміст повідомлення починається з команди "/play". Якщо так, він переходить до обробки посилання YouTube та потенційного відтворення аудіо.

– const youtubeLink = extractYouTubeLink(message.content) виклик функції extractYouTubeLink, щоб витягти посилання YouTube з вмісту повідомлення.

– if (!youtubeLink) перевіряє, чи дійсний youtubeLink. Якщо він null або порожній, це означає, що користувач не надав дійсного посилання YouTube. Бот надсилає повідомлення про помилку та виходить із команди.

– const videoInfo = await getVideoInfo(youtubeLink) виклик асинхронної функції getVideoInfo, щоб отримати інформацію про відео YouTube, таку як його назву.

– if (!videoInfo) блок перевірки videoInfo на null або порожній об’єкт. Якщо це так, то існує проблема отримання інформації про контент. Бот надсилає повідомлення про помилку

– const TguildId = "1222338279555465276" цу рядок, що визначає ID сервера, де бот повинен відтворювати аудіо. Це наразі жорстко закодовано і його слід замінити динамічним способом визначення гільдії.

– const guild = bot.guilds.cache.get(TguildId) отримання об’єкту Discord-гільдії за допомогою наведеного ID. Він використовує метод .cache.get для доступу до кешованих гільдій.

Код створення з’язку з голосовим каналом (лістинг 3.7).

Лістинг 3.7 – Друга частина блоку bot.on(Events.MessageCreate)

```
const connection = joinVoiceChannel(
  {
    channelId: channelID,
    guildId: TguildId,
    adapterCreator: message.guild.voiceAdapterCreator
  });
console.log(connection)
const player = createAudioPlayer();
connection.subscribe(player);
```

Кінець лістингу 3.7

– const connection = joinVoiceChannel({...}) приєднання до вказаного голосового каналу та встановлення голосового з’єднання. Він приймає об’єкт з наступними властивостями:

– channelId: Це ID голосового каналу, до якого потрібно приєднатися. У вашому коді channelID, ймовірно, є змінною, яка містить ID каналу, отриманий з голосового каналу користувача або іншого призначеного каналу.

– guildId: Це ID сервера, до якої належить канал.

– adapterCreator: Це функція, яка створює аудіоадаптер для голосового з’єднання.

– const player = createAudioPlayer() це рядок, що створює об’єкт аудіоплеєра за допомогою функції createAudioPlayer з бібліотеки Discord.js. Аудіоплеєр буде відповідати за відтворення потоку аудіо.

– connection.subscribe(player) підключає створений player до connection, встановленого на попередньому кроці. Це гарантує, що потік аудіо з плеєра буде надсилатися через голосове з’єднання до голосового каналу.

Створення аудіоресурсу, програвання його, та реалізація команди «стоп» (лістинг 3.8).

Лістинг 3.8 – Третя частина bot.on(Events.MessageCreate)

```
const resource = createAudioResource(ytdl(youtubeLink, { filter: "audioonly" }), { inlineVolume: true });
        resource.volume.setVolume(1);
        player.play(resource);
        player.once(AudioPlayerStatus.Idle, async () => {
          connection.destroy();
          player.destroy();
        });
        message.channel.send(`Now playing: ${videoInfo.title}`);
      }
      if (message.content.startsWith("/stop")) {
        connection.destroy();
        player.destroy();
      }
  });
```

Кінець лістингу 3.8

– const resource = createAudioResource(ytdl(youtubeLink, { filter: "audioonly" }), { inlineVolume: true }) це рядок, що створює об’єкт аудіоресурсу за допомогою функції createAudioResource з бібліотеки Discord.js. Цей ресурс буде використовуватися для представлення потоку аудіо, який буде відтворено. Першим аргументом createAudioResource є результат роботи функції ytdl. Ця функція приймає посилання YouTube та отримує потік аудіо. Опція filter: "audioonly" гарантує, що це буде лише аудіо. Другим аргументом createAudioResource є об’єкт з додатковими властивостями. inlineVolume: true вказує, що гучністю можна керувати безпосередньо на об’єкті ресурсу.

– resource.volume.setVolume(1) це рядок, що встановлює початкову гучність аудіоресурсу на повну гучність.

– player.play(resource) це рядок, що розпочинає відтворення аудіоресурсу за допомогою методу player.play. Поток аудіо буде надсилатися через голосове з'єднання, встановлене раніше.

– player.once(AudioPlayerStatus.Idle, async () => {...}) це рядок, що налаштовує прослуховувач подій для події AudioPlayerStatus.Idle. Ця подія буде генеруватися, коли відтворення аудіо буде завершено або зупинено. Функція зворотного виклику в фігурних дужках буде виконуватися асинхронно, коли подія буде спрацьована. Всередині функції зворотного виклику ви можете виконувати дії очищення після завершення відтворення. У цьому випадку він знищує об'єкти connection та player, щоб звільнити ресурси.

– message.channel.send(Now playing: ${videoInfo.title}) це рядок, що надсилає повідомлення в канал, де було видано команду, вказуючи, що відео YouTube зараз відтворюється. Він використовує властивість videoInfo.title, отриману раніше, щоб відобразити назву відео.

– if (message.content.startsWith("/stop")) це блок, що перевіряє, чи вміст повідомлення починається з команди "/stop". Якщо так, він продовжує зупиняти відтворення.

– connection.destroy() це рядок, що знищує голосове з'єднання, фактично від'єднуючи бота від голосового каналу.

– player.destroy() це рядок знищує об'єкт аудіоплеєра, зупиняючи будь-яке поточне відтворення.

3.2 Реалізація мобільного додатку

Структура мобільного додатку комплексніша, ніж в діскорд бота, проте, причина цьому базові налаштування проекту, яке створює базову структуру з додатковими ресурсами. Огляд структури проекту на рисунку 3.8.

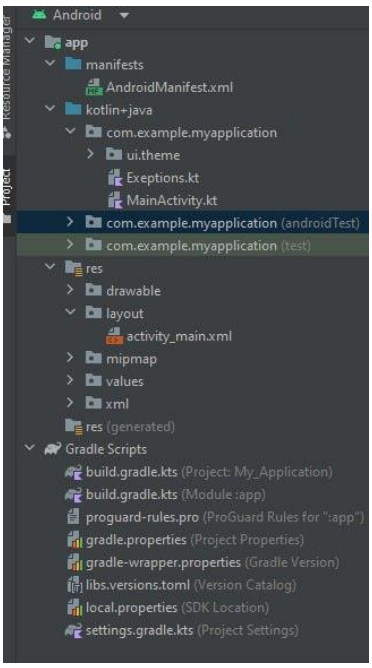

Рисунок 3.8 – структура мобільного додатку

Для реалізації користувацького інтерфейсу додатку використовувся Layout View та програмована генерація елементів інтерфейсу. Базовий інтерфейс додатку показаний на рисунку 3.9.

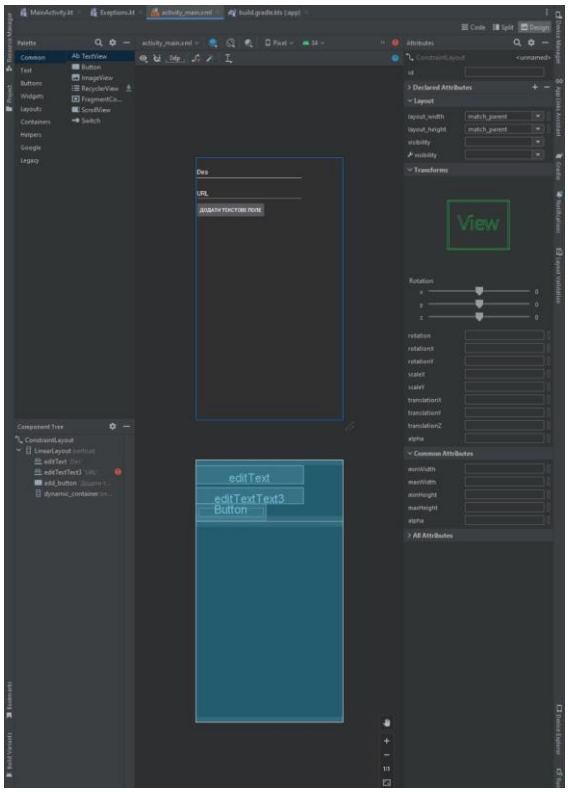

Рисунок 3.9 – базовий інтерфейс додатку

Файл MainActivity.kt містить основні функції додатку. Для реалізації функціоналу списку, як інструменту для відправки запитів, потрібно прописати комплексну функцію, що створює об’єкт для відправки, та встановлює з’єднання з вебхуком. Розроблено функцію fetchDataInBackground (лістинг 3.9).

Лістинг 3.9 – функція fetchDataInBackground

```
private fun fetchDataInBackground(link: String, id: String) {
    val data = JSONObject()
    data.put("content", "/play " + link)
    data.put("id", id)
    val jsonString = data.toString()
    val contentType = "application/json; charset=utf-8".toMediaTypeOrNull()
    val client = OkHttpClient()
    val gson = Gson()
    val loggingInterceptor = HttpLoggingInterceptor()
      .setLevel(HttpLoggingInterceptor.Level.BODY)
    val request = Request.Builder()
      .url("https://discord.com/api/webhooks/1242940570427457536/hua4uCeP0-
HRnR3rOEYCkyYtCQk4KRvekzc03MV2K-j9xptNGEz3hn31tlxmc6OCfmEA")
      .post(jsonString.toRequestBody(contentType))
      .build()
    val okHttpClient = OkHttpClient.Builder()
      .addInterceptor(loggingInterceptor)
      .build()
    val backgroundThread = Thread {
      try {
        val response = okHttpClient.newCall(request).execute()
        if (response.isSuccessful) {
          val responseCode = response.code
          val mainHandler = Handler(Looper.getMainLooper())
          mainHandler.post {
            // Update UI here based on successful response
            println("Request successful with code: $responseCode")
          }
        } else {
          val exception: Exception = when (response.code) {
            404 -> PageNotFoundException()
            else -> IllegalStateException("Error")
          }
          throw exception
        }
      } catch (e: Exception) {
        if (e is NetworkException) {
```

Продовження лістингу 3.9

```
                println("Network message " + e.message)
            } else {
                println(e.message)
            }
        }
    }
    backgroundThread.start()
}
```

Кінець лістингу 3.9

Створюється об’єкт JSON, що містить інформацію про запит (команда "/play", ID запиту). Дані JSON перетворюються на текстовий формат.

Визначається тип контенту запиту. Створюються об’єкти для HTTP-клієнта, обробника JSON та перехоплювача журналювання. Створюється HTTP-запит з URL-адресою вебхука Discord, методом POST та тілом JSON. 3. Виконання запиту у фоновому потоці: Створюється новий потік для виконання запиту, щоб не блокувати головний потік. У блоці try-catch: Виконується HTTP-запит. Перевіряється успішність запиту.

Для реалізації зіставлення списків аудіо для конференцій, потрібно створити динамічний список. Для цього реалізовано функцію addTextField (лістинг 3.10).

Лістинг 3.10 – функція addTextField

```
private fun addTextField(tag: String, link: String) {
    val textView = TextView(this)
    textView.id = View.generateViewId()
    textView.text = tag
    textView.setOnClickListener {
        fetchDataInBackground(link, "137947417910837248")
    }
    val layoutParams = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
    textView.layoutParams = layoutParams
    dynamicContainer.addView(textView)
}
```

Кінець лістингу 3.10

– val textView = TextView(this) створює новий об’єкт TextView, пов’язаний з поточним контекстом активності (this).

– textView.id = View.generateViewId() призначає текстовому полю унікальний ідентифікатор за допомогою View.generateViewId(). Це важливо для майбутньої взаємодії з конкретним текстовим полем.

– textView.text = tag: встановлює текст, який відображається текстовим полем, у рядок tag, що надається. Ця зміння зберігає посилання на контент.

– textView.setOnClickListener встановлює прослуховувач клацання для текстового поля. Коли користувач клацає на текстове поле, код у фігурних дужках буде виконано.

– fetchDataInBackground(link, "137947417910837248"): Викликає попередньо визначену функцію fetchDataInBackground, передаючи наданий link та жорстко закодований рядок ідентифікатора текстового чату «137947417910837248».

– val layoutParams = LinearLayout.LayoutParams(...) створює новий об’єкт параметрів макета типу LinearLayout.LayoutParams. Цей об’єкт визначає, як текстове поле буде розміщено в його батьківському макеті

– layoutParams.width = LinearLayout.LayoutParams.MATCH_PARENT встановлює ширину текстового поля, щоб вона відповідала ширині його батьківського макета.

– layoutParams.height = LinearLayout.LayoutParams.WRAP_CONTENT: встановлює висоту текстового поля, щоб воно обгортало його вміст.

– textView.layoutParams = layoutParams застосовує створені параметри макета до текстового поля.

– dynamicContainer.addView(textView) додає новостворене текстове поле з його прослуховувачем клацання та параметрами макета до dynamicContainer, для створення динамічної верстки.

В основному тілі функції onCreate, що є частиною життєвого циклу додатку Anroid, виконуються та використовуються вищеперечислені функції, та реалізовано логіку для комунікації з базами даних (лістинг 3.11).

Лістинг 3.11 – основне тіло додатку

```
    val userInputLink: EditText = findViewById(R.id.editTextText3)
    val config = HikariConfig().apply {
        driverClassName = "org.mariadb.jdbc.Driver"
       jdbcUrl = "jdbc:mariadb://192.168.0.103:3306/link_list"
        username = "root"
        password = "18891889"
        connectionTimeout = 2000
        maximumPoolSize = 10
    }
    val dataSource = HikariDataSource(config)
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val userInputTag: EditText = findViewById(R.id.editText)
    userInputLink.setText("")
    addButton = findViewById(R.id.add_button)
    dynamicContainer = findViewById(R.id.dynamic_container)
    addButton.setOnClickListener {
        val tag = userInputTag.text.toString()
        val link = userInputLink.text.toString()
        addTextField(tag, link)
    }
        val data = fetchDataFromDatabase(dataSource)
        runOnUiThread {
            // Update UI on main thread after data is fetched
            for (entry in data) {
                addTextField(entry.tag, entry.link)
            }
        }
    }.start()
}
```

Кінець лістингу 3.11

Створює пул з’єднань з базою даних MariaDB. Визначає URL-адресу, ім’я користувача та пароль для доступу до бази даних. Завантажує макет інтерфейсу користувача. Знаходить елементи керування, такі як поля вводу, кнопка «Додати» та контейнер для посилань. Встановлює початковий текст для полів вводу. Додає прослуховувач натискання до кнопки «Додати». При натисканні кнопки: Отримує текст з полів вводу для тега та посилання. Додає новий запис до списку з отриманим тегом і посиланням. Отримання даних з бази даних: Створює фоновий потік для витягу даних з бази даних. Отримує дані з бази даних. Оновлює UI в головному потоці, додаючи нові записи до списку на основі отриманих даних.

3.3 Розробка бази даних та тестування проекту

Була розроблена невелика база данних на базі MariaDB. Складалась вона з одної таблиці з трьома стовпцями: ID елементу, його назви, та самого посилання (рис. 3.10). Сессія налаштована на IP локальної машини, та був встановлений вільний порт 3306 (рис. 3.11).

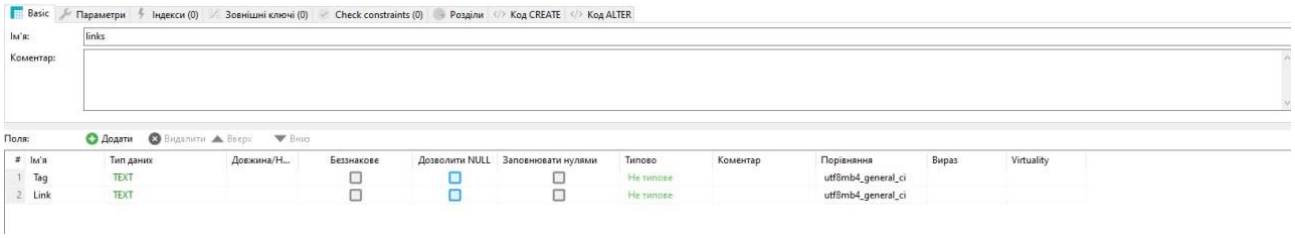

Рисунок 3.10 – Структура таблиці в базі данних

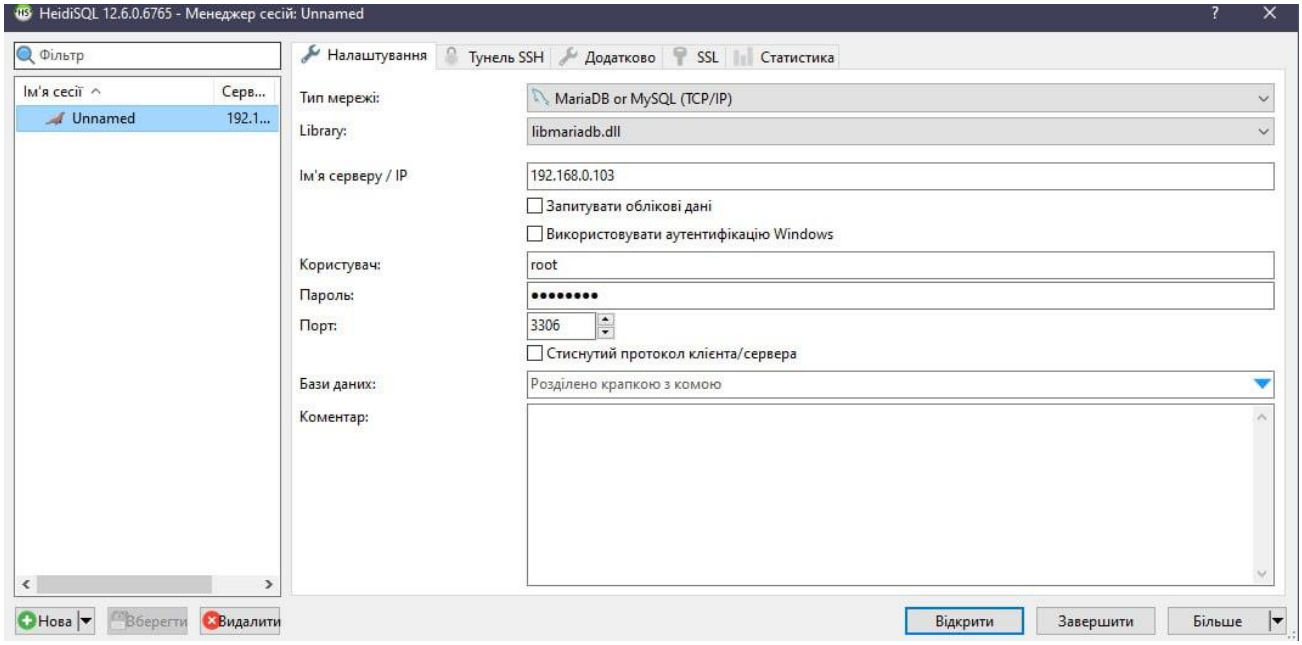

Рисунок 3.11 – Налаштування сессії

В базі данних знаходиться один елемент, а саме Vid, та посилання на відео (рисунок 3.12). Як можна замітити на рисунку 3.13, в списку додатку підтягнулась інформація з бази даних. Слідуючи з рисунка 3.14, функція поповнення списку також працює.

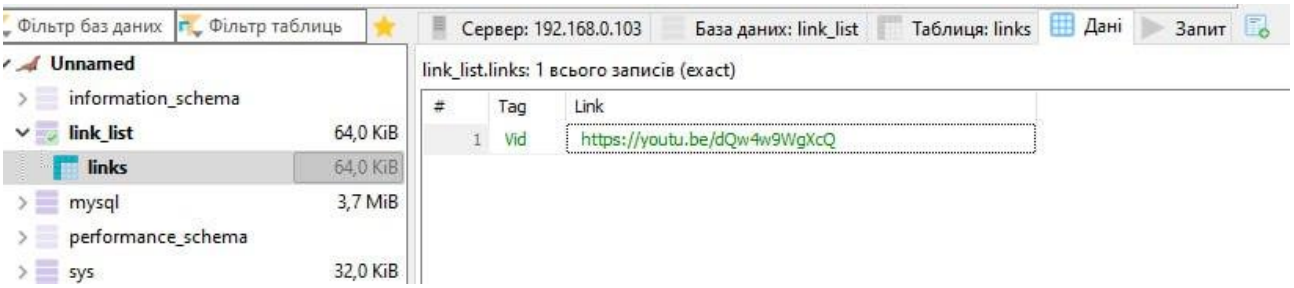

Рисунок 3.12 – Вміст таблиці links

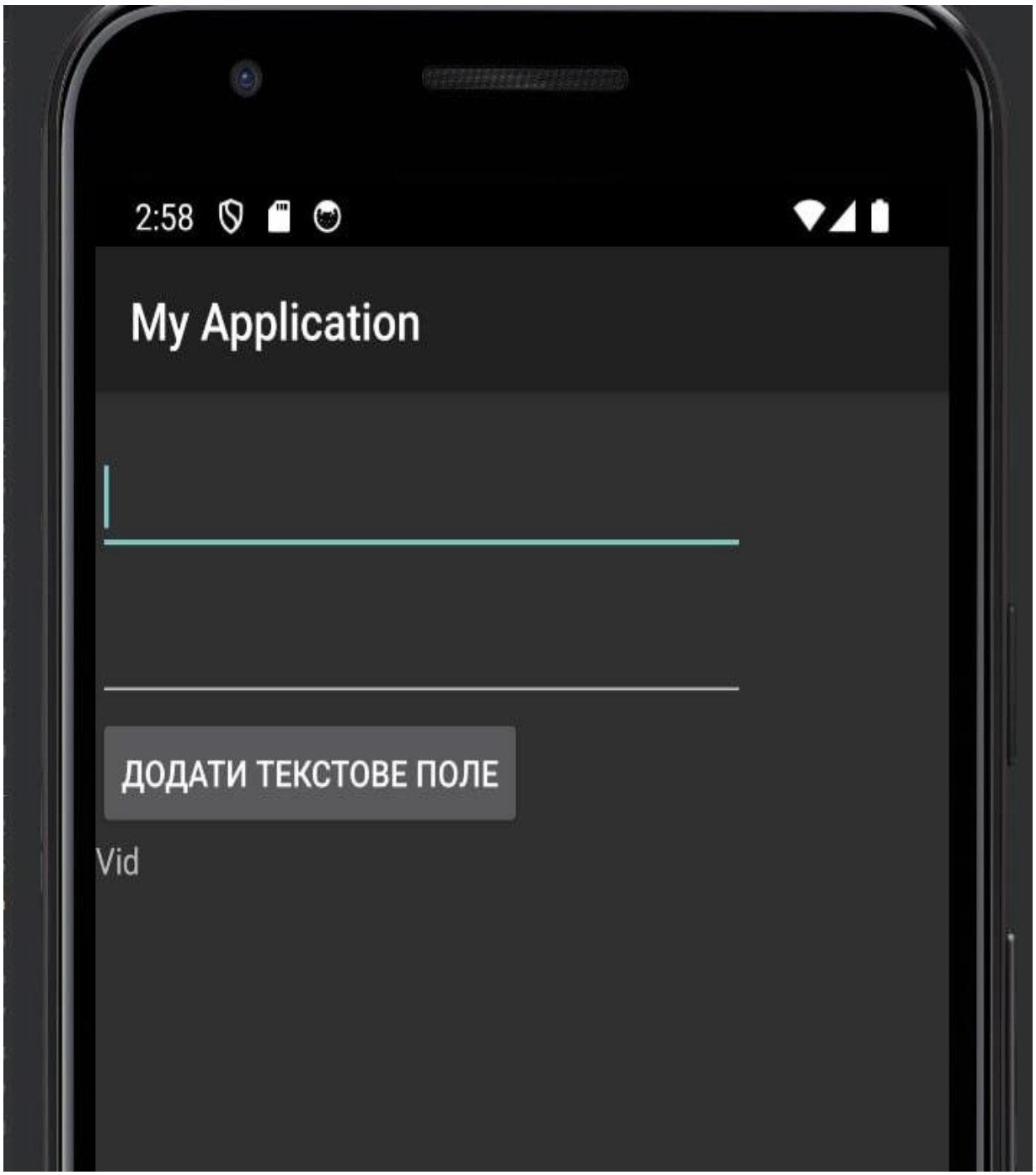

Рисунок 3.13 – Додаток при запуску

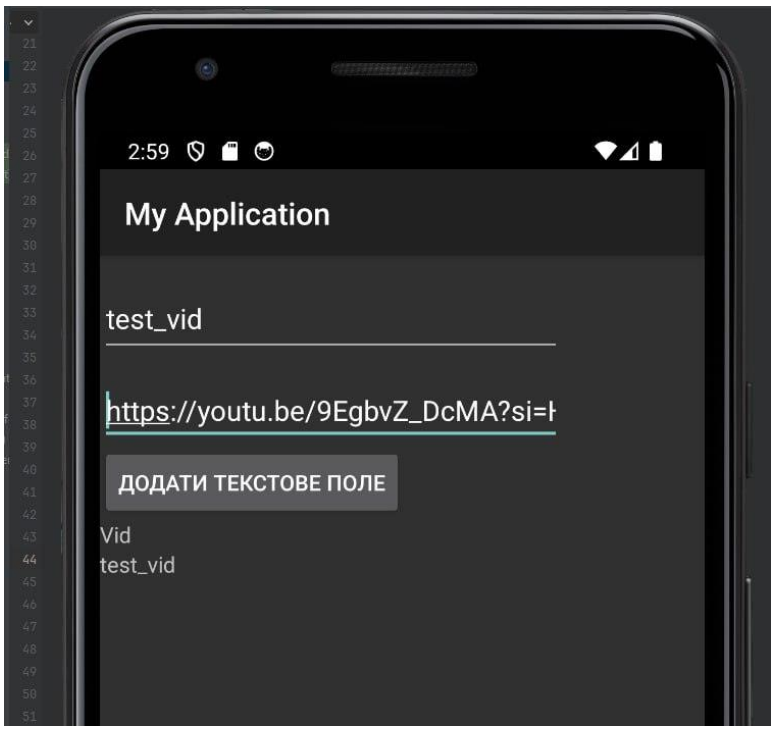

Рисунок 3.14 – Демонстрація функції поповнення списку

А також, згідно рисунку 3.15, можемо побачити, що після кліку на останній елемент списку, вебхук «Capitain Hook» пересилає запит в чат з мобільного додатку у вигляді сповіщення – «/play https://youtu.be/9EgbvZ_DcMA?si=Hhx...». Бот обробляє команду, написану вебхуком, та додається до голосового чату для програвання аудіо, отриманого з відео.

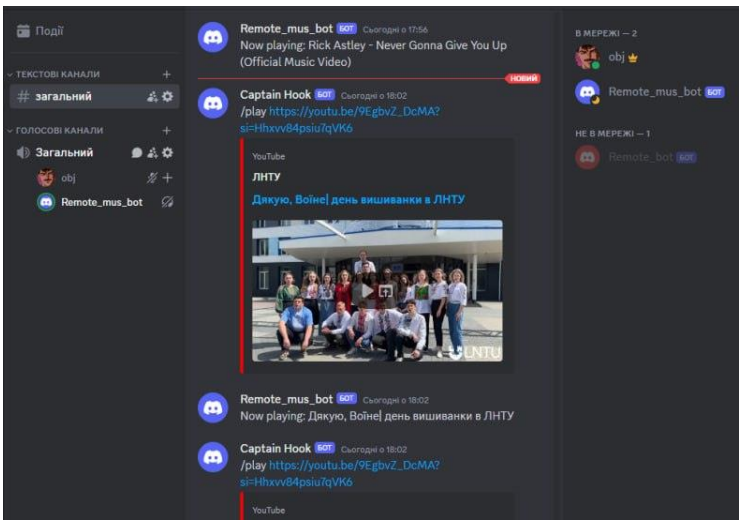

Рисунок 3.15 – Демонтрація роботи Discord бота

ВИСНОВКИ

В ході дослідження була розроблена система додатків для оформлення онлайн зустрічей, в яку входить Discord бот, мобільний Android додаток , вебхук, для реалізації комунікації між ними, а також база даних для зберігання інформації.

Проаналізовані найпоширеніші методи взаємодії між Discord сервером, та ботами, а саме: командна, контекстна, інтерактивна та гібридна взаємодії. Також було розглянуто види ботів за їхнім використанням.

Були реалізовані функції для переводу даних, та їхнього надсилання, отримання даних з зовнішніх джерел(YouTube), керування подіями за допомогою Discord API, динамічне оновлення верстки додатку, а також взаємодія з зовнішніми базами даних та отримання інформації.

В ході аналізу інструментів та бібліотек були відібрані найоптималіні рішення, а саме: використання JavaScript та Node.JS для реалізації діскорд бота, ytdl та discord.js як основні модулі для бота, розробки додатку на базі Android Studio, та okhttp, GSON та HikariCP як основних бібліотек для реалізації потрібних нам функцій мобільного додатку.

Також, використання Raspberry PI стало гарною моделлю веб сервера, особливо в цьому випадку, коли даних не так багато та навантаження на сервер не велике. Важко аналізувати роботу Raspberry PI в таких невеликих масштабах, проте як веб-сервер для домашнього проекту вона чудово себе показала.

Присутні недоліки та можливі покращення подібної системи. Серед них мушу відмітити засмічення текстового чату, що можна вирішити за допомогою згаданого в роботі скрипта посередника, проблеми з безпекою, що можна вирішити за допомогою імплементації користувацького вводу ID текстових та голосових чатів, а також можливість оновлювати базу даних лиш мануально, що вирішується імплементацією логіки відправки наявного, оновленого списку в додатку до бази даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ETHW. Picturephone. ETHW. URL: https://ethw.org/Picturephone (дата звернення: 10.05.2024).

2. CUSeeMe activity. Service – Faculty Websites. URL: https://web .cortland.edu/flteach/methods/obj2/cuseeme2.html (дата звернення: 10.05.2024).

3. Introducing the webex app. Cisco. URL: https://www.cisco.com/c/en/us/ solutions/collaboration/webex-call-message-meet.html#~solution (дата звернення: 10.05.2024).

4. Cóndor-Herrera O., Ramos C., Janio Jadán-Guerrero J. Analysis of video conferencing platforms for online teaching during the Covid-19 pandemic. ResearchGate. URL: https://www.researchgate.net /publication/361247192_Analysis_ of_video_conferencing_platforms_for_online_teaching_during_the_Covid-19_pandemic (дата звернення: 10.05.2024).

5. TeamSpeak vs. discord: choose the best option. Trusted Multimedia Software for your Mac. URL: https://www.movavi.com/learning-portal/teamspeak-vs-discord.html (date of access: 11.05.2024).

6. What is Discord: A Guide for Parents and Educators. Discord. URL: https://discord.com/safety/360044149331-what-is-discord (дата звернення:: 13.05.2024).

7. Basic Channel Setup. Discord. URL: https://discord.com /community/basic-channel-setup (date of access: 13.05.2024).

8. Discord API docs for bots and developers. Discord Developer Portal. URL: https://discord.com/developers/docs/reference (дата звернення: 17.05.2024).

9. Discord bots. Discord bot list. URL: https://discordbotlist.com/ (дата звернення: 17.05.2024).

10. Вступ до JavaScript. Сучасний підручник з JavaScript. URL: https://uk.javascript.info/intro (дата звернення: 17.05.2024).

11. Node.js – about node.js®. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/en/about (дата звернення: 17.05.2024).

12. Discord.js. URL: https://discord.js.org/docs/packages/discord.js/14.15.3 (дата звернення: 18.05.2024).

13. PHP: what is PHP? – manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/en/intro-whatis.php (дата звернення: 18.05.2024).

14. ytpl. npm. URL: https://www.npmjs.com/package/ytpl (дата звернення: 18.05.2024).

15. Discord developer portal API docs for bots and developers. Discord Developer Portal. URL: https://discord.com/developers/docs/resources/webhook (дата звернення: 22.05.2024).

16. What is android studio? definition from techtarget. Mobile Computing. URL: https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio#:~:text=Android%20Studio%20is%20the%20official,code%20editing%20and%20developer%20tools. (дата звернення: 22.05.2024).

17. Kotlin for android. Kotlin Help. URL: https://kotlinlang.org/ docs/android-overview.html (дата звернення: 22.05.2024).

18. Overview – OkHttp. Square Open Source. URL: https://square.github.io/ okhttp/ (дата звернення: 23.05.2024).

19. Gson – quick guide. Online Tutorials, Courses, and eBooks Library |Tutorialspoint. URL: https://www.tutorialspoint.com/gson/gson_quick_guide.htm (дата звернення: 23.05.2024).

20. Introduction to HikariCP Baeldung. URL: https://www.baeldung.com/ hikaricp (дата звернення: 23.05.2024).

21. About MariaDB Connector. MariaDB KnowledgeBase. URL: https://mariadb .com/kb/en/about-mariadb-connector-j/ (дата звернення: 23.05.2024).

22. MariaDB server documentation. MariaDB KnowledgeBase. URL: https://mariadb.com/kb/en/documentation/ (дата звернення: 25.05.2024).

23. HeidiSQL – MariaDB/MySQL, MSSQL, PostgreSQL, SQLite and Interbase/Firebird made easy. URL: https://www.heidisql.com/ (дата звернення: 25.05.2024).

24. Raspberry Pi 4 Model B review. Tom’s Guide. URL: https://www.tomsguide.com/reviews/raspberry-pi-4-model-b (дата звернення: 25.05.2024).